Vision and Cognitive Systems Report

Mahir Selek
mahir.selek@studenti.unipd.it

Abstract

DeepDream is a highly experimental technique and there is no one "right" way to use it. It often requires a lot of experimentation and trial-and-error to produce interesting and visually appealing results

1. Introduction

Deep learning (DL) has the ability to achieve complicated cognitive tasks exceeding the performance of humans. Because the performance of DL algorithms, such as convolutional neural network(CNN), gives great results compared to other machine learning machine learning.

This project analyzes that implementation of Deep Dream on different architectures. In this application, how outputs are obtained due to the structural differences of different architectures will be examined in detail.

1.1. Motivations Behind the Study

Comparing different pre-trained models in a Deep Dream project can help us understand how each model is capturing different features and structures of the image. Each model has its own architecture and was trained on same datasets, so they will have different strengths and weaknesses.

By comparing the output of each model, we can see which model is more unique at capturing the features that we are interested in, and which model produces more interesting and visually appealing results.

1.2. Overview of Models and Results

In order to get an idea on the outputs, 3 different models were applied, which are Inception-V3, VGG19 and ResNet. In addition, Imagenet dataset was used to train these models as only one image is processed.

Among Inception-V3, ResNet, and VGG19, VGG19 is considered to be the simplest architecture. It has a sequential structure with 19 layers and uses only 3x3 convolutional layers and 2x2 pooling layers. In contrast, Inception V3 and ResNet have more complex structures with skip connections and various types of convolutional layers. However, simplicity of architecture does not necessarily mean inferior performance. Each architecture has its strengths and weaknesses and may perform better or worse depending on the specific task. Personally, the Inception V3 model revealed more obvious visual patterns than the others.However, I should point out that the ResNet model produced the unique random shapes among them.

2. Related Works

This section presents some of the most directly and closely related works to deep dream. I did minimal implementation of this blog post by Alexander Mordvintsev. Different from this minimal study, different pre-trained models with different level of hyperparameters were applied in this project.

Relatedly,in this application, the DeepDream minimal project was implemented directly with the Inception v3 model.

3. Dataset

A direct dataset was not used in this project, since it was implemented over a single image, instead a pre-trained imagenet dataset was used when initializing the models.

The ImageNet dataset [6] is a large-scale visual recognition challenge dataset containing millions of labeled images. The models which are used in this project was pretrained on this dataset and learned to classify images into one of 1,000 categories.

3.1. Aim to use Imagenet

Figure 1. ImageNet

By using these pre-trained weights, the model that we use can quickly and accurately identify features in new images that may be useful for generating visually interesting deepdream images.

This is because the models which is pre-trained has already learned to recognize patterns and features that are commonly present in real-world images, which can help guide the deepdream algorithm to generate interesting and meaningful images.

In the given code, weights='imagenet' indicates that the pre-trained weights of the models are initialized using the weights that were trained on the ImageNet dataset.

3.2. Image used in the project

This image was chosen as the most suitable for the project among different images from the Pinterest site.

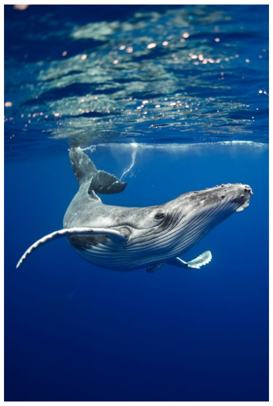


Figure 2. Whale Image

4. Models

Pre-trained models are machine learning models that have been trained on large datasets for specific tasks, such as image classification or object detection.These models are typically trained on massive datasets, such as ImageNet, which contains millions of labeled images as we did in this project. The models use a technique called transfer learning, which means that the pre-trained models have already learned to recognize patterns and features that are commonly present in real-world images.

Using pre-trained models can be very beneficial for several reasons:

- Saves time and resources,
- Improved accuracy,
- Easier to develop.

The aim of using pre-trained models like InceptionV3, ResNet, and VGG19 in the DeepDream project is to leverage the deep learning capabilities of these models to generate interesting and meaningful images that have similar patterns and features. Also we can take advantage of the feature-extraction and recognition capabilities of these models.

4.1. Inception-V3 Model

The inception v3 model was released in the year 2015, it has a total of 42 layers and a lower error rate than its predecessors [8].

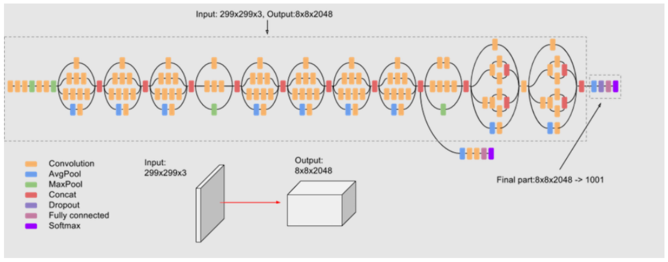
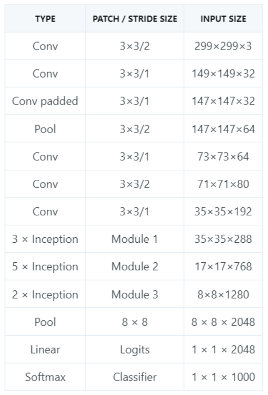


Figure 3. InceptionV3 Architecture

Firstly, one of the major assets of the Inception V3 model is the generous dimension reduction.It means that the reduced number of parameters the computational costs also reduce. In total, the inception V3 model is made up of 42 layers which is a bit higher than the previous inception V1 and V2 models.We can see in detail what are the components the Inception V3 model is made of.



TYPE	PATCH / STRIDE SIZE	INPUT SIZE
Conv	3×3/2	299×299×3
Conv	3×3/1	149×149×32
Conv padded	3×3/1	147×147×32
Pool	3×3/2	147×147×64
Conv	3×3/1	73×73×64
Conv	3×3/2	71×71×80
Conv	3×3/1	35×35×192
3 × Inception	Module 1	35×35×288
5 × Inception	Module 2	17×17×768
2 × Inception	Module 3	8×8×1280
Pool	8 × 8	8 × 8 × 2048
Linear	Logits	1 × 1 × 2048
Softmax	Classifier	1 × 1 × 1000

Figure 4. InceptionV3 Structure

The above table describes the outline of the inception V3 model. Here, the output size of each module is the input size of the next module.

The architecture of the Inception-v3 model consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. One key feature of Inception-v3 is its use of "inception modules", which are composed of multiple convolutions with different kernel sizes and pooling operations. These modules allow the network to extract features at multiple scales, which is important for recognizing objects of different sizes in an image.

The output of each convolutional layer is fed into a pooling layer, which down-samples the feature map by taking the maximum or average value within a certain window. This helps reduce the size of the feature map and also makes the network more invariant to small spatial shifts in the input image. During training, the network's weights are adjusted using backpropagation and gradient descent to minimize a loss function that compares the predicted class probabilities to the true class labels.

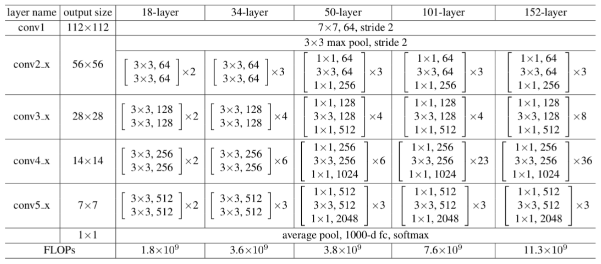
Overall, the Inception-v3 architecture is designed to be both accurate and efficient, with a relatively small number of parameters compared to other state-of-the-art models at the time of its introduction.

4.2. ResNet50 Model

ResNet50 is a deep neural network architecture that was proposed by Microsoft Research in 2015. It is a variant of the ResNet architecture [5], which stands for Residual Network, and is designed to allow for the training of very deep neural networks.

The ResNet50 architecture consists of 50 layers and is designed to learn the mapping between the input image and the output class label. It has a few unique features that set it apart from other architectures, including:

- Skip connections,
- Convolutional layers,
- Global average pooling
- Batch normalization



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 5. ResNet50 Architecture

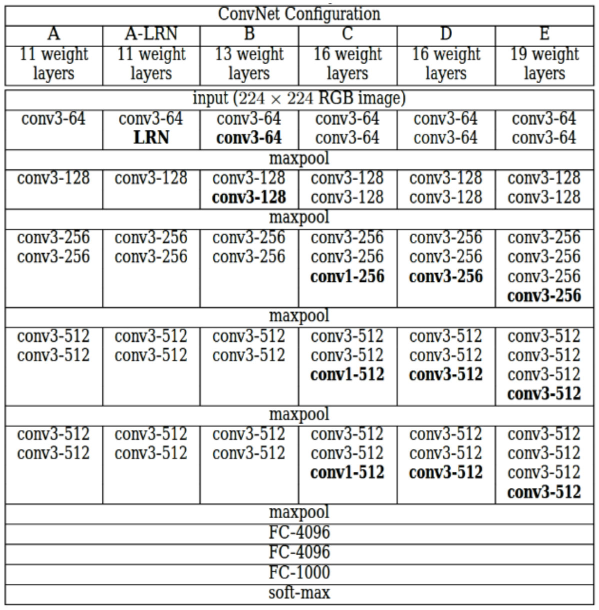
ResNet introduced a new architecture that includes residual connections, which allow information to be directly passed from one layer to another without going through the intermediate layers. This helps to alleviate the vanishing gradient problem that can occur in very deep networks and enables ResNet to be trained to much greater depths than Inception V3 and VGG.In this paper,the authors report that ResNet outperforms Inception V3 and VGG on the ImageNet dataset for large-scale image recognition tasks [5].

However, it is worth noting that the choice of architecture depends on the specific task at hand, and there is no one-size-fits-all solution. In some cases, Inception V3 or VGG may be more appropriate depending on the data and the task.

4.3. VGG19 Model

VGG19 is a convolutional neural network model that was developed by the Visual Geometry Group at the University of Oxford[7]. It is part of a family of VGG models, which were created to achieve high accuracy on image classification tasks. VGG19 is a deep neural network that contains 19 layers, including 16 convolutional layers, 3 fully connected layers, and max pooling layers. It has over 140 million parameters, making it a very powerful model.

One of the main features of VGG19 is its use of small 3x3 filters in the convolutional layers. This allows for a larger receptive field and more complex feature extraction. The architecture of VGG19 is also relatively simple and easy to understand, with a series of convolutional layers followed by max pooling layers, which downsample the spatial dimensions of the output.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 6. VGG19 Architecture

Compared to Inception V3 and ResNet50, VGG19 has a simpler architecture with smaller kernel sizes and fewer convolutional layers. This makes it more computationally efficient to train and use. Additionally, VGG19 has been shown to perform well on image recognition tasks, especially when the images are not too complex, such as in the case of ImageNet.

However, for more complex tasks with larger and more varied images, Inception V3 and ResNet50 may be more suitable due to their more advanced architectures and ability to handle more complex features.

5. Experiments

This experiment is based on the models that we defined before. The DeepDream algorithm utilizes the concept of octaves to analyze the input image at multiple scales through an iterative process. With each octave, the input image is downsampled and then analyzed by the neural network to extract patterns and features specific to that particular scale. This allows for the creation of a series of images, each capturing unique features and details of the original image at a different scale.

Good results were achieved after the first soft octave part but there were still some issues to be resolved. The difference is examined in detail below.

5.1. Implementation

The main idea in DeepDream is to choose a layer (or layers) and maximize the ”loss” in a way that the image increasingly ”excites” the layers. The complexity of the features incorporated depends on layers chosen by us, i.e, lower layers produce strokes or simple patterns, while deeper layers give sophisticated features in images, or even whole objects.

Using different layers will result in different dream-like images. Deeper layers respond to higher-level features (such as eyes and faces), while earlier layers respond to simpler features (such as edges, shapes, and textures).

This can be thought of as asking the network “Based on what you already know, can you see anything here that you recognise?”.

5.2. Preprocessing

Basically, in this project, we normalize the input image to ensure that its pixel values fall within a certain range that is suitable for the deep learning model being used.

After that downsizing process applied. Downsizing the image makes it easier to work with. Also, without donwsizing tried in this project, but encountered errors and waste of time during the learning phases.

5.3. Model creation

The model creation started with using the module in Tensorflow’s Keras library [2] to create a new instance(s) of the each model.

The weights parameter is set to ’imagenet’, which means that the model is initialized with pre-trained weights on the ImageNet dataset. This is useful because the model can then identify and extract features from new images with a high degree of accuracy.

5.4. Loss Calculation

In DeepDream, the loss is calculated as the sum of activations in the selected layers of the neural network. To prevent larger layers from dominating the contribution to the loss, normalization is applied to each layer.Normally, loss is a quantity you wish to minimize via gradient descent. In DeepDream, you will maximize this loss via gradient ascent.

5.5. Gradient Ascent

Once we have calculated the loss for the chosen layers, all that is left is to calculate the gradients with respect to the image, and add them to the original image.Adding the gradients to the image enhances the patterns seen by the network.

In other words, we do classical gradient descent optimization with some differences :

- The weights of the model are freezed, so the model doesn’t get updated.
- The parameters to optimize are the pixels of the initial image.
- The loss function is the score of some given class.
- We do maximization instead of minimization.

5.6. DeepDream Main Loop

The main loop initializes the variables and enters a loop that will repeatedly apply the DeepDream algorithm to the input image until the desired number of steps have been completed. Within the loop, it correctly handles the case where the remaining number of steps is less than 100, and updates the `steps_remaining` and `step` variables accordingly. The loss and modified image are then printed and displayed using the show function. Once the DeepDream process is complete, the modified image is de-processed and returned as the final result.

The weights parameter is set to ’imagenet’, which means that the model is initialized with pre-trained weights on the ImageNet dataset. This is useful because the model can then identify and extract features from new images with a high degree of accuracy.

After the Main Loop outputs, the results were not very satisfactory and there were no overly pronounced patterns.

One approach that addresses all these problems is applying gradient ascent at different scales. This will allow patterns generated at smaller scales to be incorporated into patterns at higher scales and filled in with additional detail.

To do this you can perform the previous gradient ascent approach, then increase the size of the image (which is referred to as an octave), and repeat this process for multiple octaves

We will discuss the octaving process in detail in the next section. We will also see how it has significant consequences on outputs.

5.7. DeepDream with Octave

The original approach towards octave convolutions has been introduced by [1] [3].

In the ”octave” version of DeepDream, the input image is processed at multiple scales, with each scale (or octave) capturing different levels of detail.The technique works by maximizing the activations in certain layers of the network.

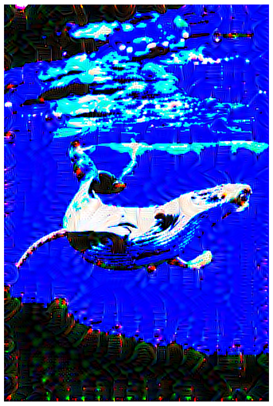


Figure 7. VGG-DeepDream

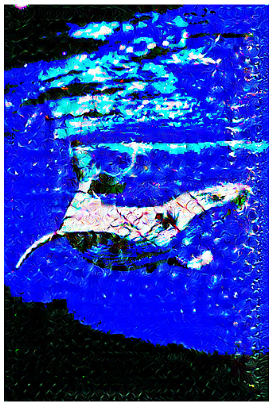


Figure 8. ResNet-DeepDream

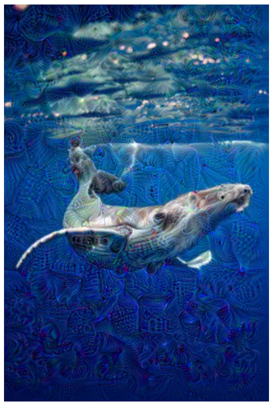


Figure 9. InceptionV3-DeepDream

5.8. Scaling up with tiles

One thing to consider is that as the image increases in size, so will the time and memory necessary to perform the gradient calculation. The above octave implementation will not work on very large images, or many octaves.

To avoid this issue the image splitted into tiles and compute the gradient for each tile [4].

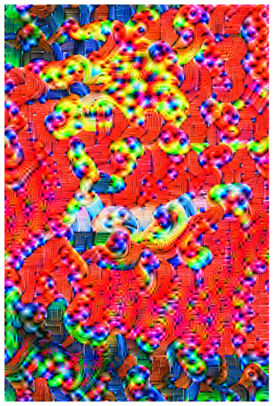


Figure 10. VGG-Scale Up

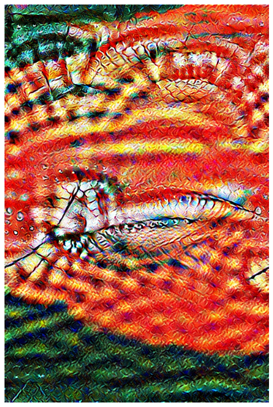


Figure 11. ResNet-Scale Up

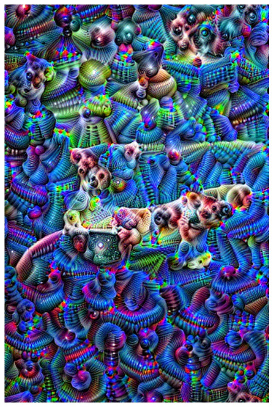


Figure 12. InceptionV3-Scale Up

6. Conclusion

The DeepDream project is a fascinating application of neural networks that allows us to create visually stunning images by maximizing the activations in certain layers of a pre-trained convolutional neural network.

In this project, we used the InceptionV3, ResNet50, and vGG19 models to generate these images, and we explored different techniques to make the images more diverse and interesting, such as using different scales, adding color, and combining multiple layers. We also used a technique called ”Scale Up” to gain a deeper understanding of what the network is learning at each layer, and we discovered that some layers appear to specialize in detecting specific features, such as edges, textures, or eyes.

Overall, the project demonstrates the power and flexibility of convolutional neural networks, and how they can be used not only for traditional classification tasks but also for creative applications like DeepDream. It also highlights the importance of pre-trained models and how they can be leveraged to perform complex tasks with relatively little data or computational resources.

References

[1] Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3435–3444, 2019.

[2] François Chollet. Introduction to keras. *March 9th*, 2018.

[3] Ricard Durall, Franz-Josef Pfreundt, and Janis Keuper. Stabilizing gans with soft octave convolutions. *arXiv preprint arXiv:1905.12534*, 2019.

[4] Amirata Ghorbani and James Y Zou. Neuron shapley: Discovering the responsible neurons. *Advances in Neural Information Processing Systems*, 33:5922–5932, 2020.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.